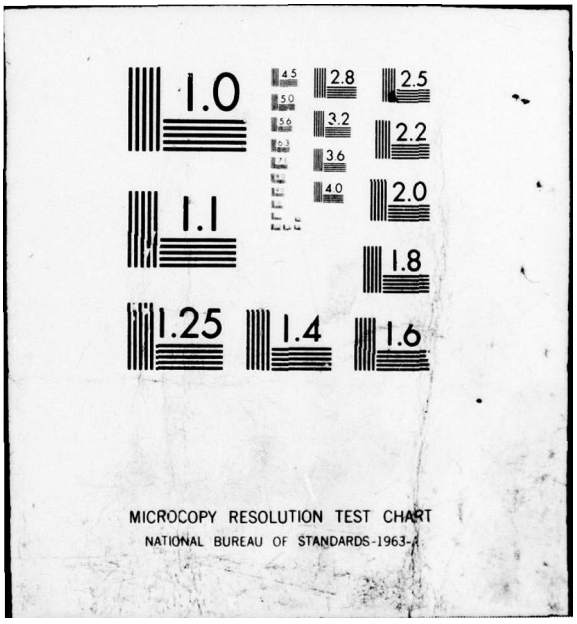
AD-A065 209 ARIZONA UNIV TUCSON DEPT OF COMPUTER SCIENCE F/G 9/2
TRANSFERRING FILES TO AN IBM MACHINE.(U)
JUL 78 R J ORGASS AFOSR-78-3499
UNCLASSIFIED TM-APLAD14 AFOSR-TR-79-0127 NL

| OF |
AD
A065209

END
DATE
FILMED
4-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

THE UNIVERSITY OF ARIZONA
TUCSON, ARIZONA 85721

DEPARTMENT OF COMPUTER SCIENCE

(6) TRANSFERRING FILES TO AN IBM MACHINE*

(10) Richard J. Orgass

(7) Technical Memorandum No. APLAD14

(11) 28 July 28, 1978

(12) 8p. (14) TM-APLAD14

ABSTRACT

Directions for using a sequence of programs to write EBCDIC or ASCII files that can be read by IBM machines and other computers is described. The programs have been used to transfer files to an IBM machine and to a HP-2000.

* (15) Research sponsored by the Air Force Office of Scientific Research, Air Force Systems Command, under Grant No. AFOSR-78-3499. The United States Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation hereon.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFOSR-TR- 79-0127	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)*: TRANSFERRING FILES TO AN IBM MACHINE	5. TYPE OF REPORT & PERIOD COVERED: Interim
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Richard J. Orgass	8. CONTRACT OR GRANT NUMBER(s): AFOSR 78-3499
9. PERFORMING ORGANIZATION NAME AND ADDRESS: The University of Arizona, Department of Computer Science, Tucson, Arizona 85721	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Office of Scientific Research/NM, Bolling AFB, Washington, DC 20332	12. REPORT DATE: July 28, 1978
	13. NUMBER OF PAGES: 7
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*: UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

ACCESSION for
NTIS White Section X
DDC Buff Section
UNANNOUNCED
JUSTIFICATION
BY
DISTRIBUTION/AVAILABILITY CODES
Dist. AVAIL. and/or SPECIAL
A

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Directions for using a sequence of programs to write EBCDIC or ASCII files that can be read by IBM machines and other computers is described. The programs have been used to transfer files to an IBM machine and to a HP-2000.

DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

1. Introduction

The programs described here were written in order to make it possible for me to move my research software to VPI. Since I assembled the programs, they have been used by others so I decided to write documentation in case they are useful to others.

I make no claims about the reliability of these programs other than to report that tapes written following this procedure have been successfully read on an IBM 370/158 and on a HP-2000. Beyond this, use them at your own risk.

When moving files to an IBM machine, it's important to realize that two things we take for granted in the DEC-10 world don't work in reasonable ways. (1) Each file has a record length associated with it. Each record in the file must contain exactly this many characters and each line is padded with blanks to be the correct length. (2) Tabs are not available to reduce the number of characters in a file. Before transmitting files, the tabs that we regularly use must be replaced by blanks.

The directions for using this set of programs is rather complicated; it was planned as a single application and not as a production system.

There are four steps to be completed to write a tape: (1) Preparation of the files. (2) Creating a report on the record lengths of the files. (3) Generating a command file for CHANGE. (4) Writing a tape using CHANGE.

In the discussion that follows, I will use MUMBLE as a file name with several extensions. When you run the programs, replace all occurreces of MUMBLE with a file name of your choice.

2. Preparing Files for Transmission

The first step in this sequence of events is to collect all of the files that are to be written on a tape in a single directory. SFD's must be expanded. After you have collected the files, some editing can save pain later.

For source program files, it's a good idea to add a new first line which is a comment containing the file name, for example, in ALGOL:

```
COMMENT file APL0.SIM;
```

This makes it easier to identify files if there is confusion at the other end.

The EBCDIC character set has one annoying property: There are no square brackets! Files going to an IBM machine should have square brackets replaced with the appropriate characters for the IBM implementation. For SIMULA, the square brackets are replaced by parentheses; an ugly substitution! [Incidentially, when writing EBCDIC, CHANGE converts left square brackets into underscores (_) and right square brackets into single quotes (').]

If you are transferring SIMULA programs, the following changes are sufficient: (1) Replace the characters \= with the four characters *NE* . (2) Replace all occurrences of the character \ with the five characters *NOT* . (3) If you used the DEC-10 specific comment character !, replace all occurrences of this character with the keyword *COMMENT*. Programs transformed in this way have been successfully compiled using the IBM SIMULA compiler; changes to calls to class infile are also needed for execution.

After you have edited your files as appropriate, it is time to remove line numbers and to expand the tabs. This can be done with the following:

```
.r pip
*all:*.*/w/x/n=all:*.*
*↑C
```

At this point, you are ready to analyze your files to determine the record lengths that are needed.

2. Computing Record Lengths and Reporting

The first thing to do is to create a file that contains a directory of your files. Here is a good way to do this:

```
.r aplsf
TERMINAL..KEY
APL-10 DEC+SYSTEM+10+APL+SF E1(30)
TTY47) ...
CLEAR WS
      FOO←⍵')LIB *.*'
      FOO←FOO[⍋FOO;]
      )OUTPUT MUMBLE.DIR
      FOO
      )OUTPUT
      )MON
```

Here, the characters typed with the APL ball are in the APL character set, not ASCII!

The file specification in the)*LIB* command may be any legal DEC-10 file specification with scan and wild. Of course, structures may be specified and ALL: is the default.

There may well be other ways to make a directory file but the output format of this procedure is expected by later programs.

The next step is to modify the directory file to remove two spurious lines. The following will do the job:

```
.so mumble.dir
Editing STDN:MUMBLE.DIR
*d↑
*d*
*eun
[STDN:MUMBLE.DIR]
```

This file, MUMBLE.DIR, is input to the program that computes record lengths by counting characters in each record of the file. Here is the terminal transcript:

```
.r csc:lrecl
LRECL -- Version 1, 5/31/78.
File List Input File: MUMBLE.DIR
Report Output file: MUMBLE.RPT
```

When LRECL is in execution, an output file is written and the same report is printed on the terminal to provide some assurance that computation is in progress. There is one line of output for each file. This line contains the file name followed by 5 numbers: (1) The length of the longest record in the file. (2) The number of records in the file. (3) The number of characters in the file. (4) The number of characters in a file with all records padded with blanks to be as long as the longest record. (5) The number of characters in a file with all records padded with blanks to be as long as the maximum of 80 or the length of the longest record.

Files with extensions APL, REL, ATR, EXE, SAV, HGH, and LOW are viewed as binary files and their report line will have all zeros. These files will <u>not</u> be written on tape!

The last line of output gives the totals for columns (2) to (5) and is followed by the message End of SAIL execution.

The file MUMBLE.RPT that has been created is the input to the next step. I also find it useful to print the directory and report files:

```
.pri mumble.dir,mumble.rpt
```

4. Writing a Command File

The next step is to write a command file for the program CHANGE. It is possible to write either EBCDIC or ASCII files for reading on an IBM machine. There are two different directions, given below.

It is possible to write ASCII files and then convert them to EBCDIC at the destination IBM machine. This has the advantage that the translation will be done in accord with the local translation from ASCII terminal characters to EBCDIC. The disadvantage is that not all IBM installations support ASCII. To be safe, write EBCDIC but if you can find out if your destination machine supports ASCII this is better because CHANGE does some funny translations.

4.1 Writing EBCDIC Tapes

For EBCDIC, comands to write 9-track EBCDIC at 1600 bpi are generated. An APL workspace is used to write the command file. Here is a transcript:

```
.r aplsf
TERMINAL..KEY
 ....
 ....
CLEAR WS
     )LOAD CSC:CONVER
SAVED ...
COMMAND FILE GENERATOR.
INPUT FILE: MUMBLE.RPT
OUTPUT FILE: MUMBLE.CMD
NUMBER OF COPIES ON TAPE: 5
```

The response to the last query is the number of copies of each file that is to be written on the tape. In this case, 5 repitions of all the files will be written on the tape. That is, all of the files named in MUMBLE.DIR will be written, then this writing will repeat 4 more times.

When the program is running, each of your file names will be written followed by a cumulative total of the number of feet of tape that are required. This number is

computed by counting the number of characters to be written and computing the length of tape to write this many characters. In addition, one inch is allowed for the inter record gap. It appears to be a reliable estimate.

After all of the files are reported, the command file is written and the message

COMMAND FILE CREATED.

is written on the terminal. After this, return to the monitor with the command:

)MON

At this point, you may want to change the number of copies written on the tape. Just before the command *)MON*, enter the line

CONVERT∆COMMANDS

and then go back to the beginning of this section.

4.2 Writing ASCII Tapes

For ASCII, commands to write 9-track, 800 bpi tapes are generated. The files contain 8-bit ASCII (with leading bit 0) and all files are padded to the length of the longest record. There are no carriage return - line feed pairs between lines!

The terminal transcript is almost the same as in 4.1. The initial segment, which is different follows:

```
.r aplsf
TERMINAL..KEY
  ...
  ...
CLEAR WS
     )LOAD CSC:CONV2
```

Notice that the single difference is that another workspace is loaded!

5.Writing the Tape

Finally, it's time to submit a batch job to write the tape you wanted in the first place. All that is needed is a very simple CTL file. They are described below.

5.1 Writing EBCDIC Tapes

Using your favorite editor, create a file named MUMBLE.CTL with the following contents:

```
.mount m9:mtape/reel:xxxxx/den:1600/we:yyyyy
.r pub:change
*help
*mumble.cmd@
*exit
.dismount mtape:
.r summary
```

Here, xxxxx is the reel number of the tape you are going to write and yyyyy is the password you have assigned to the tape. Note that both the tape name and the density must match this mount command. If you want a different density, replace all occurrences of 1600 by your density in the text of the function *CONVERT&COMMANDS* before following the directions in Section 4.

At last, submit the file:

```
.submit mumble.ctl/time:05:00
```

The listing in the LOG file will contain a complete description of each file and the form in which it is written on the tape. The HELP file that is printed before the individual file descriptions provides enough information to understand the log.

5.2 Writing ASCII Tapes

Using your favorite editor, create a file named MUMBLE.CTL with the following contents:

```
.mount m9:mtape/reel:xxxxx/den:800/we:yyyyy
.r pub:change
*help
*mumble.cmd@
*exit
.dismount mtape:
.r summary
```

Here, xxxxx is the reel number of the tape you are going to write and yyyyy is the password you have assigned to the tape. Note that both the tape name and the density must match this mount command. If you want a different density, replace all occurrences of 800 by your density in the text of the function *CONVERT&COMMANDS* before following the directions in Section 4. Submit the CTL file as in 5.1.